Semiannual Status Report

NAG 5-1612

Submitted to
Instrument Division
Engineering Directorate
Goddard Space Flight Center
Greenbelt, MD 20771
Attn: Pen-Shu Yeh

K. Sayood, Y.C. Chen, S. Nori, and A. Araj

Department of Electrical Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0511

Period:June 15, 1993 - December 15, 1993

1 Introduction

The work performed in this period (along with work in preceeding periods) is reported in a PhD Dissertation entitled *Video Transmission on ATM Networks* by Y-.C. Chen, and a paper presented at the *International Telecommunication Conference.* We give a brief description of the work below, with details left to the dissertation and paper which are included with this report. Also included is a paper on constrained joint source/channel coding submitted to the *IEEE Journal on Selected Areas of Communications.*

Currently we are continuing with work in both areas reported here. In particular, we are examining a number of strategies for error concealment in packet video, and extending the vector quantization work.

2 Video Transmission on ATM Networks

The emergence of broadband ISDN as the network for the future brings with it the promise of integration of all proposed services in a flexible environment. In order to achieve this flexibility, asynchronous transfer mode (ATM) has been proposed as the transfer technique. This ATM-based B-ISDN network will be the carrier for services like HDTV, interactive television, multimedia workstation, and a lot more. As can be seen from these applications the proposed network will carry a lot of video information. In the past, video coding algorithms have been developed mainly for use over dedicated communication links. Although many advantages can be foreseen from this new environment, video transmission over ATM networks also present serious challenges to network providers and video specialists. These include the development of fair and efficient resource allocation schemes, policing functions, and more importantly, from the point of view of the video specialist, the bridging of network- based performance parameters and controls to the corresponding entities in video coding.

During this period we conducted a study on the bridging of network transmission performance and video coding. In the ideal case one would have have real time simulators for both video codec and network. The interactions between these two elements could then be studied extensively. However, it

would require a huge amount of effort to build a real time simulator. Given the fact that there is still a lot of uncertainty about proposed video coding algorithms and network protocols, building a simulator that would handle all the different scenarios is not feasible. The approach taken in this work is to deal with each key component in packet video separately. By doing that, we hope to obtain an in-depth understanding of the whole problem and come up with suitable solutions.

The successful transmission of variable bit rate video over ATM networks relies on the interaction between the video coding algorithm and the ATM networks. Two aspects of networks that determine the efficiency of video transmission are the resource allocation algorithm and the congestion control algorithm.

The resource allocation algorithm dictates the cost and blocking probability of a connection depending on the traffic's characteristic. An efficient resource allocation scheme increases network utilization and therefore decreases the cost of transmission. A promising approach to resource allocation is *equivalent bandwidth allocation* [1]. This approach not only describes the required bandwidth for different traffic scenarios based on traffic characteristics and quality of service(QOS) requirements but is easy to manage as well.

The congestion control algorithm is a major factor in determining the quality of a call. The policing function plays a vital role in monitoring traffic flow and thus maintains a well-operated network situation. Unfortunately, because of the variety of traffic, it is not an easy task to effectively regulate connection to its agreed-upon contract effectively. Of the schemes proposed to date the *leaky bucket(LB)* algorithm comes closest to being effective. We propose a *dual leaky bucket* mechanism based on equivalent bandwidth assignment, with the first bucket monitoring the mean bandwidth and the second one monitoring the equivalent bandwidth. With such a design, a misbehaved connection can be easily detected and network congestion can be prevented effectively (if resource allocation is performed appropriately). Also network utilization is effective with a good resource allocation scheme which takes advantage of multiplexing gain. Other congestion control approaches which have effects in video codec design will also be investigated.

Based on the understanding of the transmitting channel, we propose a complete set of design principles for video codecs. Closely following the concept

of the dual leaky bucket mechanism, a prioritized coding scheme is presented and its performance is studied. We also develop some combined approaches to smooth the video output flow. This in turn leads to a reduction in the requested equivalent bandwidth. Finally, some error control algorithms are proposed to combat the effect of cell loss which comes from the nature of packet video.

There are still a lot of issues about B-ISDN left to be clarified. It will require extensive efforts in order to clear the confusion among user, service provider, and equipment manufacturer and accelerate the pace of implementing B-ISDN. This work provides a design approach for video transmission based on the understanding and evaluation of current ATM networks. We also hope that the results and conclusions presented in this work may contribute to create a guideline for the design of packet video codec in the future.

3 Vector Quantization for Nonstationary Sources

Introduction

Vector quantization (VQ) is one of the more popular compression techniques to appear in the last twenty years. Numerous compression techniques, which incorporate VQ, have been proposed. While the LBG VQ [2] provides excellent compression, there are also several drawbacks to the use of the LBG quantizers. These include search complexity and memory requirements, especially at higher rates, and a mismatch between the codebook and the inputs. The latter mainly stems from the fact that the VQ is generally designed for a specific rate and a specific class of inputs. When the bandwidth constraints and/or the source statistics change this can result in severe degradation in the quality of the reconstructed output.

In order to reduce the search complexity, a number of techniques have been proposed which impose structure on the codebook entries. These include tree structured VQs, lattice VQs, and classified VQs [3]. However, each approach has its own drawbacks. The tree structured and classified VQs do nothing about the memory requirements. In fact the tree structured VQs can actually excarbate the memory requirements problem. The lattice quantizers

can avoid both search and memory problems, however, these quantizers lack the pattern matching ability of the LBG VQ. Residual quantizers (RQs), also known as multiple-stage VQs, have been introduced to reduce both the computation and memory requirements [4, 5, 6].

All of these methods assume that the code book, large or small, accurately reflects the input image statistics. However this is not always the case, especially when coding a nonstationary source such as a video sequence. Various approaches which use some sort of codebook adaptation have been proposed in which elements of the codebook are replaced as coding proceeds [7, 8, 9, 10]. These approaches are generally of a forward adaptive nature in that the update procedure requires the transmission of side information.

In this work, we propose an adaptive technique for vector quantization of images and video sequences. The technique is an extension of the recursively indexed scalar quantization (RISQ) algorithm [11]. This approach involves the use of a small code book, reducing the computational complexity. The code book adapts to the input statistics. We present both forward and backward adaptation rules.

Proposed Quantizer

While the RISQ algorithm has been quite successful in a number of applications, it is not possible to directly extend it to vector quantization as there are some fundamental differences between scalar and vector quantizers. The input to a scalar quantizer is assumed to be *iid*. The vector quantizer on the other hand can be viewed as a pattern matching algorithm [12]. The input is assumed to be one of a number of different patterns. The scalar quantizer is used after the redundancy has been removed from the source sequence, while the VQ takes advantage of the redundancy in the data.

With these differences in mind we view the recursively indexed vector quantizer (RIVQ) as a two stage process. The first stage performs the normal pattern matching function, while the second stage recursively quantizes the residual if the magnitude of the residual is greater than some prespecified threshold. The codebook of the second stage is ordered so that the magnitude of the codebook entries is a nondecreasing function of its index. We then choose an index I which will determine the mode in which the RIVQ

operates.

The quantization rule $\mathcal{Q}$ is given as follows:

For a given input value x_0, we have the following:

- Quantize x_0 with the first stage quantizer Q_1.
- If the residual $||x_0 - Q_1(x_0)||$ is below a specified threshold then $Q_1(x_0)$ is the nearest output level.
- Otherwise generate $x_1 = x_0$ - $Q_1(x_0)$ and quantize using the second stage quantizer Q_2. Check if the index J_1 of the output is below the index I. If so, $Q(x_0) = Q_1(x_0) + Q_2(x_1)$. If not, form $x_2 = x_2 - Q(x_1)$ and do the same for the same as for x_1.

This process is repeated until for some time m, the index J_m falls below the index I, in which case x_0 will be quantized to

$$\mathcal{Q}(x_0) = Q_1(x_0) + Q_2(x_1) + ... + Q_2(x_M).$$

Thus, the RIVQ operates in two modes: it operates in one mode when the index J of the quantized input falls below a given index I and another when the index J falls above the index I.

Methods for updating the code book

In this section, we present two algorithms used to update the first stage quantizer. The adaptation algorithms use the fact that with the RIVQ the output values are always within a prescribed distance of the inputs. This means that the set of output values of the RIVQ can be viewed as an accurate representation of the inputs and their statistics. In the following we first present a backward adaptive algorithm which uses only the outputs for adaptation, and an adaptation algorithm which uses some side information for adaptation. We have called the algorithm forward adaptive even though this algorithm also uses the past outputs for adaptation.

Backward Adaptive Quantization In this method, we divide the input sequence into intervals and use the outputs of quantizer of the previous interval as a training sequence and the present code book as initial code book for the generalized Lloyd algorithm. By clustering all the past quantized outputs, a new code book is generated. This new code book is used to encode the next block of input. We note that in the beginning the training sequence is small depending on the length of the update interval, but after some time, the length of the training sequence increases. The second stage VQ could also be updated in a similar fashion. In this method no overhead is needed since both the encoder and the decoder generate the same code book.

Forward Adaptive Quantization In this approach we treat a subset of the output set of the previous intervals as our codebook. We use the method described in [13] to inform the receiver of which elements of the previous outputs form the codebook for the next interval. Suppose an output set, in order of first appearance, is $\{p, a, q, s, l, t, r\}$, and the desired codebook for the interval to be encoded is $\{a, q, l, r\}$, then we would transmit the binary string0110101 to the receiver. The 1s correspond to the letters in the output set which would be elements of the desired codebook. We select the subset for the current interval by finding the closest vectors from our collection of past outputs to the input vectors of the current set. This means that there is an inherent delay of one interval imposed by this approach. The overhead required to send the codebook selection is M/N where M is the number of vectors in the output set, and N is the interval size.

Preliminary Simulation Results

We simulated the proposed technique using the *Lena* image and the *Xray, Couple* and *Girl* images from the USC database. The images were divided into 4×4 blocks (dimension 16). The initial code books for both stages were generated using the *USC Girl* and the *USC Couple* images as the training set. The threshold was chosen to be 500. The update interval was every 8192 pixels or 512 vectors.

We compared the results of the proposed systems with an LBG VQ with the test image outside the training set and where the test image was also

the training image. As expected the proposed approach significantly outperformed the LBG VQ with the mismatched training image. However even for the case where the LBG VQ used the same image as both training and test image, the performance of the proposed system was very close to the LBG VQ. We are currently using the proposed technique to code video sequences which we will compare to the MPEG algorithm.

In all cases that the adaptation is very robust, and the performance close to the ideal omniscient case. Therefore the technique could be used in situations where the source statistics are unknown or change rapidly. We intend to present results further justifying these conclusions by the time of the conference.

References

[1] R. Guérin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and its Application in High Speed Networks. *IEEE J. Selected Areas Communications*, 9:968–981, September 1991.

[2] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantization design. *IEEE Transactions on Communications*, COM-28:84–95, Jan. 1980.

[3] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1991.

[4] B.H. Juang and A.H. Gray. Multiple Stage Vector Quantization for Speech Coding. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 597–600. IEEE, April 1982.

[5] C.F. Barnes and R.L. Frost. Residual Vector Quantizers with Jointly Optimized Code Books. In *Advances in Electronics and Electron Physics*, pages 1–59. Elsevier, 1992.

[6] C.F. Barnes and R.L. Frost. Vector Quantizers with Direct Sum Codebooks. *IEEE Transactions on Information Theory*, 39:565–580, March 1993.

[7] D. Paul. A 500-800 bps Adaptive Vector Quantization Vocoder Using a Perceptually Motivated Distortion Measure. In *Conference Record, IEEE Globecom*, pages pp. 1079–1082. IEEE, 1982.

[8] A. Gersho and M. Yano. Adaptive Vector Quantization by Progressive Codevector Replacement. In *Proceedings ICASSP*. IEEE, 1985.

[9] M. Goldberg and H. Sun. Image Sequence Coding ... *IEEE Transactions on Communications*, COM-34:703–710, July 1986.

[10] O.T-C. Chen, Z. Zhang, and B.J. Shen. An Adaptive High-Speed Lossy Data Compression. In *Proc. Data Compression Conference '92*, pages 349–355. IEEE, 1992.

[11] K. Sayood and S. Na. Recursively Indexed Quantization of Memoryless Sources. *IEEE Transactions on Information Theory*, IT-38, November 1992.

[12] A. Gersho and V. Cuperman. A Pattern Matching Technique for Speech Coding. *IEEE Communications Magazine*, pages 15–21, December 1983.

[13] S. Panchanathan and M. Goldberg. Adaptive Algorithm for Image Coding Using Vector Quantization. *Signal Processing: Image Communication*, 4:81–92, 1991.

APPENDIX

Vector Quantization of Non Stationary Sources*

Ali G. Al-Araj and Khalid Sayood
Department of Electrical Engineering
University of Nebraska
Lincoln, NE 68588-0511

Abstract

Common problems with Vector Quantization (VQ) include encoding complexity, memory requirements, a mismatch between training and test statistics, and 'overload' errors. These problems become increasingly pronounced when dealing with nonstationary sources such as a video source. We propose an adaptive vector quantization algorithm which uses an extension of the recursively indexed scalar quantizer to resolve these problems. The use of a recursively indexed VQ results in distortion limited outputs which can be used in adaptive algorithms. We present a backward adaptive algorithm and another which could be classified as forward adaptive.

1 Introduction

Vector quantization (VQ) is one of the more popular compression techniques to appear in the last twenty years. Numerous compression techniques, which incorporate VQ, have been proposed. The basic technique is simple. The source sequence is blocked into vectors. These vectors are compared to a set of representative vectors called a codebook. The index of the vector in the codebook which provides the closest match (generally in the sense of the L_1 or L_2 norm) is then transmitted. The codebook is usually generated using a clustering algorithm. The most popular algorithm for codebook generation is the generalized Lloyd algorithm proposed by Linde, Buzo, and Gray [1]. There are several drawbacks to the use of the LBG quantizers. These include search complexity and memory requirements, especially at higher rates, and a mismatch between the codebook and the inputs. The latter mainly stems from the fact that the VQ is generally designed for a specific rate and a specific class of inputs. When the bandwidth constraints and/or the source statistics change this can result in severe degradation in the quality of the reconstructed output.

There have been a number of attempts at overcoming these problems. In order to reduce the search complexity, a number of techniques have been proposed which impose structure on the codebook entries. These include tree structured VQs, lattice VQs, and classified VQs [2]. However, each approach has its own drawbacks. The tree structured and classified VQs do nothing about the memory requirements. In fact the tree structured VQs can actually excarbate the memory requirements problem. The lattice quantizers can avoid both search and memory problems, however, these quantizers remove no redundancy from the source.

Residual quantizers (RQs), also known as multiple-stage VQs, have been introduced to reduce both the computation and memory requirements. In the RQs, each stage uses a small code book to encode the errors (residuals) of the preceding stage [3]. Recently, Barnes and Frost [4, 5] investigated the use of direct-sum code book with RQs to minimize the memory requirements of VQs. In their work, the RQ stages are jointly optimized. They concluded that their new design method led to an improvement in performance of the VQs.

All of these methods assume that the code book, large or small, accurately reflects the input image statistics. However this is not always the case, especially when coding a nonstationary source such as a video sequence. To guard against degradations in the quality of the reconstruction one could use a large "universal" codebook which was trained using vectors from a number of statistically different sources [6]. Unfortunately, due to the size requirements this can excarbate the search and memory requirements. Furthermore it is a *minimax* solution, which while providing some insurance against severe degradation, does not provide the best performance for a given codebook size. These problems can be somewhat alleviated by using only a subset of the universal codebook at any given time [7]. The composition of this sub-

*This work was supported by the NASA Goddard Space Flight Center under Grant NAG 5-1612.

set depends on local statistics and is transmitted to the decoder as side information. This avoids the minimax problem while retaining the "universality" of the codebook. Other approaches which use some sort of codebook adaptation have also been proposed in which elements of the codebook are replaced as coding proceeds [8, 9, 10, 11]. These approaches are generally of a forward adaptive nature in that the update procedure requires the transmission of side information.

In this paper, we propose an adaptive technique for vector quantization of nonstationary sequences. As our application area we use image coding. The technique is an extension of the recursively indexed scalar quantization algorithm [12]. This approach involves the use of a small code book, reducing the computational complexity. The code book adapts to the input statistics. We present both forward and backward adaptation rules.

This paper is organized as follows. First, the basic technique involving the estension of the RISQ algorithm is introduced in section 2. In section 3, methods used to update the code book are presented . This is followed in section 4 by some preliminary simulation results.

2 The Recursively Indexed Vector Quantizer (RIVQ)

In [12] a Recursively Indexed scalar quantizer (RISQ) was presented. The RISQ algorithm is briefly described as follows.

For a given quantizer stepsize Δ and a positive integer K, define x_l and x_h as follows:

$$x_l = -\lfloor \frac{K-1}{2} \rfloor \Delta$$

$$x_h = x_l + (K-1)\Delta$$

where $\lfloor x \rfloor$ is the largest integer not exceeding x. A recursively indexed quantizer of size K is a uniform quantizer with step size Δ (the uniform spacing both between the thresholds and between the output levels) and with x_l and x_h being its smallest and largest output levels (Q defined this way always has 0 as an output level). The quantization rule Q is given as follows:

For a given input value x if x falls in the interval $(x_l + (\Delta/2), x_h - (\Delta/2))$, then $Q(x)$ is the nearest output level. If x is greater than $x_h - (\Delta/2)$, see if

$x_1 \stackrel{\Delta}{=} x - x_h \in (x_l + (\Delta/2), x_h - (\Delta/2))$.

If so, $Q(x) = (x_h, Q(x_1))$.

If not, form $x_2 = x - 2x_h$ and do the same as for x_1.

This process continues until for some m, $x_m = x - mx_h$ falls in $(x_l(\Delta/2), x_h - (\Delta/2))$, in which case x will be quantized into

$$Q(x) = (\underbrace{x_h, x_h, \cdots, x_h}, Q(x_m))$$

If x is smaller than $x_l + (\Delta/2)$, a similar procedure to this is used, i.e., $x_m = x - mx_l$ is formed so that it falls in $(x_l + (\Delta/2), x_h - (\Delta/2))$, and is quantized to $(x_l, x_l, \cdots, x_l, Q(x_m))$.

In summary, the quantizer operates in two modes: it operates in one mode when the input falls in the range $(x_l + \frac{\Delta}{2}, x_h - \frac{\Delta}{2})$, and another when the input falls outside of the specified range. The distortion per sample is always bounded by $\frac{\Delta}{2}$.

It is not possible to directly extend the RISQ to vector quantization as there are some fundamental differences between scalar and vector quantizers. The input to a scalar quantizer is assumed to be *iid*. The vector quantizer on the other hand can be viewed as a pattern matching algorithm [13]. The input is assumed to be one of a number of different patterns. The scalar quantizer is used after the redundancy has been removed from the source sequence, while the VQ takes advantage of the redundancy in the data.

With these differences in mind we view the recursively indexed vector quantizer (RIVQ) as a two stage process. The first stage performs the normal pattern matching function, while the second stage recursively quantizes the residual if the magnitude of the residual is greater than some prespecified threshold. The codebook of the second stage is ordered so that the magnitude of the codebook entries is a nondecreasing function of its index. We then choose an index I which will determine the mode in which the RIVQ operates.

The quantization rule Q is given as follows:

For a given input value x_0, we have the following:

- Quantize x_0 with the first stage quantizer Q_1.
- If the residual $||x_0 - Q_1(x_0)||$ is below a specified threshold then $Q_1(x_0)$ is the nearest output level.
- Otherwise generate $x_1 = x_0 - Q_1(x_0)$ and quantize using the second stage quantizer Q_2. Check if the index J_1 of the output is below the index I. If so,

$$Q(x_0) = Q_1(x_0) + Q_2(x_1).$$

If not, form

$$x_2 = x_2 - Q(x_1)$$

and do the same for the same as for x_1.

This process is repeated until for some time m, the index J_m falls below the index I, in which case x_0 will be quantized to

$$Q(x_0) = Q_1(x_0) + Q_2(x_1) + ... + Q_2(x_M).$$

Thus, the RIVQ operates in two modes: it operates in one mode when the index J of the quantized input falls below a given index I and another when the index J falls above the index I.

3 Methods for updating the code book

In this section, we present two algorithms used to update the first stage quantizer. The adaptation algorithms use the fact that with the RIVQ the output values are always within a prescribed distance of the inputs. This means that the set of output values of the RIVQ can be viewed as an accurate representation of the inputs and their statistics. In the following we first present a backward adaptive algorithm which uses only the outputs for adaptation, and an adaptation algorithm which uses some side information for adaptation. We have called the algorithm forward adaptive even though this algorithm also uses the past outputs for adaptation.

3.1 Backward Adaptive Quantization

In this method, we divide the input sequence into intervals and use the outputs of quantizer of the previous interval as a training sequence and the present code book as initial code book for the generalized Lloyd algorithm. By clustering all the past quantized outputs, a new code book is generated. This new code book is used to encode the next block of input. We note that in the beginning the training sequence is small depending on the length of the update interval, but after some time, the length of the training sequence increases. The second stage VQ could also be updated in a similar fashion. In this method no overhead is needed since both the encoder and the decoder generate the same code book.

3.2 Forward Adaptive Quantization

In this approach we treat a subset of the output set of the previous intervals as our codebook. We use the method described in [7] to inform the receiver of which elements of the previous outputs form the codebook for the next interval. Suppose an output set, in order of first appearance, is $\{p, a, q, s, l, t, r\}$, and the desired codebook for the interval to be encoded is $\{a, q, l, r\}$, then we would transmit the binary string0110101 to the receiver. The 1s correspond to the letters in the output set which would be elements of the desired codebook. We select the subset for the current interval by finding the closest vectors from our collection of past outputs to the input vectors of the current set. This means that there is an inherent delay of one interval imposed by this approach. The overhead required to send the codebook selection is M/N where M is the number of vectors in the output set, and N is the interval size.

$\|I\|$	Rate bits/pixel	# inputs to stage 2	# outputs from stage 2	PSNR
20	1.64	3378	8855	31.97
30	1.34	3590	6385	30.74
40	1.17	3590	4951	29.67

Table 1: Effect of $|I|$ on the backward adaptive RIVQ.

4 Preliminary Simulation Results

We simulated the proposed technique by applying it to the image compression problem. As our test images we used a 256 × 256 section of the Lena image and the *Xray* image from the USC database. There are several parameters in the RIVQ that can be adjusted to vary the performance. These include the threshold for quntization of the residual, the second stage index I. We do not as yet have an analytical approach to setting these parameters, and we present some initial results from empirical evaluations.

The image is divided into 4 × 4 blocks, therefore the VQs have dimension 16. The initial code books for both stages were generated using the USC *girl* and the *couple* images as the training set. The threshold was chosen to be 500. The update interval was every 8192 pixels or 512 vectors.

Table 1 shows the results for the backward adaptive approach with different values of the index I. The value shown in the table is actually the magnitude of the element in the codebook table at index I. The image used here is the a 256×256 portion of the *Lena* image. For reference a codebook designed using the Lena sub-image as both the training and test input provides a PSNR of 32.77 dB at a rate of 1.25 bits/pixel.

The difference between this and the ideal case at the same rate is around 2.5 dB. There is a distortion-

$\|I\|$	Rate	# inputs to stage 2	# outputs from stage 2	PSNR
20	1.69	3030	7342	32.46
30	1.44	3173	5312	31.42
40	1.31	3266	4280	30.49

Table 2: Effect of $|I|$ on the forward adptive RIVQ.

$\|I\|$	Rate	# inputs to stage 2	# outputs from stage 2	PSNR
30	1.30	3426	6055	31.02
40	1.15	3542	4797	30.00

Table 3: Performance of restricted forward adptive RIVQ.

rate tradeoff which is a function of I. As I increases the PSNR and the rate both drop off.

In Table 2 we look at the effect on the performance of the forward adaptive scheme when the index value I is changed.

Notice that as $|I|$ becomes larger the number of requantizations goes down, as would be expected, but the number of inputs with residual magnitudes greater than the threshold goes up. The reason for this is that as the value of $|I|$ goes up, the reconstruction accuracy goes down. This leads to codebooks that are not as representative of the input which in turn results in larger residuals.

If instead of allowing the encoder to select a codebook from the entire set of past outputs we use only the last 256 outputs as the codebook, we remove the overhead involved in codebook transmission. The results for this case are shown in Table 3.

Comparing these results to those in Table 2, we see that there has been a slight drop in rate accompanied by a slight drop in PSNR values. If we look at the number of requantizations we see that these have actually increased, so some of the savings from the overhead gets used in the requantization.

In all cases that the adaptation is very robust, and the performance close to the ideal omniscient case. Therefore the technique could be used in situations where the source statistics are unknown or change rapidly. We intend to present results further justifying these conclusions by the time of the conference.

References

[1] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantization design. *IEEE Trans. Commun.*, COM-28:84–95, Jan. 1980.

[2] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1991.

[3] B.H. Juang and A.H. Gray. Multiple Stage Vector Quantization for Speech Coding. In *Proc. ICASSP*, pages 597–600. IEEE, April 1982.

[4] C.F. Barnes and R.L. Frost. Residual Vector Quantizers with Jointly Optimized Code Books. In *Advances in Electronics and Electron Physics*, pages 1–59. Elsevier, 1992.

[5] C.F. Barnes and R.L. Frost. Vector Quantizers with Direct Sum Codebooks. *IEEE Trans. Inform. Theory*, 39:565–580, March 1993.

[6] Y. Yamada, K. Fujita, and S. Tazaki. Vector Quantization of Video Signals. In *Proceedings Annual Conference of IECE.* IECE, 1980.

[7] S. Panchanathan and M. Goldberg. Adaptive Algorithm for Image Coding Using Vector Quantization. *Signal Processing: Image Communication*, 4:81–92, 1991.

[8] D. Paul. A 500-800 bps Adaptive Vector Quantization Vocoder Using a Perceptually Motivated Distortion Measure. In *Conference Record, IEEE Globecom*, pages pp. 1079–1082. IEEE, 1982.

[9] A. Gersho and M. Yano. Adaptive Vector Quantization by Progressive Codevector Replacement. In *Proceedings ICASSP.* IEEE, 1985.

[10] M. Goldberg and H. Sun. Image Sequence Coding. *IEEE Trans. Commun.*, COM-34:703–710, July 1986.

[11] O.T-C. Chen, Z. Zhang, and B.J. Shen. An Adaptive High-Speed Lossy Data Compression. In *Proc. DCC '92*, pages 349–355. IEEE, 1992.

[12] K. Sayood and S. Na. Recursively Indexed Quantization of Memoryless Sources. *IEEE Trans. Inform. Theory*, IT-38, November 1992.

[13] A. Gersho and V. Cuperman. A Pattern Matching Technique for Speech Coding. *IEEE Communications Magazine*, pages 15–21, December 1983.

A CONSTRAINED JOINT SOURCE/CHANNEL CODER DESIGN†

Khalid Sayood*, Fuling Liu+, and Jerry D. Gibson**

*Dept. of Electrical Engineering
Univ. of Nebraska, Lincoln, Nebraska 68588-0511

+Western Atlas
Houston, TX

** Dept. of Electrical Engineering,
Texas A&M Univ., College Station, TX 77843

ABSTRACT

We examine the design of joint source/channel coders in situations where there is residual redundancy at the output of the source coder. We have previously shown that this residual redundancy can be used to provide error protection without a channel coder. In this paper we extend this approach to conventional source coder/convolutional coder combinations. We also develop a family of nonbinary encoders which more efficiently use the residual redundancy in the source coder output. We show through simulation results that the proposed systems outperform conventional source-channel coder pairs with gains of greater than 10 dB in the reconstruction signal to noise ratio at high probability of error.

†This work was supported in part by NASA Lewis Research Center (NAG 3-806) and NASA Goddard Space Flight Center (NAG 5-1612)

1 Introduction

One of Shannon's many fundamental contributions was his result that source coding and channel coding can be treated separately without any loss of performance as compared to an optimum system [1]. The basic design procedure implied by Shannon's theorems consists of designing a source encoder which changes the source sequence into a series of (approximately) independent, equally likely binary digits followed by a channel encoder which accepts binary digits and puts them into a form suitable for reliable transmission over the channel [1]. One aspect of the overall optimum system not addressed by Shannon is any increase in system complexity that results from this separation. Massey [2] and Anchetta [3] showed that for distortionless transmission of the source under the constraint of linear source and channel coders, a significant reduction in complexity with equivalent performance can be achieved using a *joint* source/channel coder. Their scheme also differs from most data compression schemes in that the bulk of the system complexity is transferred to the receiver.

The theorem that provides justification for the separate design of the source coder and the channel coder, often called the Information Transmission Theorem [1], assumes that both the source encoder/decoder pair and the channel encoder/decoder pair are operating in an optimal fashion. Specifically, the source encoder is assumed to present the channel encoder for optimal channel coding, and the channel encoder/decoder pair is assumed to reproduce the source encoder output at the source decoder input with negligible distortion. Unfortunately, there are practical situations in which these assumptions are violated – namely, when the source encoder output contains redundancy, which occurs when the source encoder is suboptimal, and when the source decoder input differs from the source encoder output, which is a result of channel errors. These two situations are common occurrences in practical communication systems where source and/or channel models are imperfectly known, complexity is a serious issue, or significant delay is not tolerable. Various approaches have been developed to handle these situations, and they are usually grouped under the general heading of *joint source/channel coding*

We have attempted to develop a more precise nomenclature by distinguishing three classes of coders. One class of coders we designate as *joint* source channel coders because the source

and channel coding operations are truly integrated, and in this class we include the work of Anchetta [3] and Massey [2], the work of Dunham and Gray [4], who proved the existence of joint source/channel trellis coders for certain fidelity criteria, and the joint source/channel coder designs of Ayanoğlu and Gray [5].

In a second class, denoted as *concatenated* source/channel coders, we place coders that cascade known source coders and known channel coders, and allocate the fixed bit rate between the source coder and the channel coder to maximize the system performance. Work in this class includes that of Modestino and Daut [6] who investigated two dimensional differential pulse code modulation (2D-DPCM) for image coding combined with short constraint length convolutional codes, Modestino, Daut, and Vickers [7] who investigated 2D discrete cosine transform (DCT) of images with convolutional codes, Modestino, Bhaskaran, and Anderson [8] who studied tree encoding of images with convolutional coding, Comstock and Gibson [9] who considered 2D-DCT coding of images in conjunction with Hamming codes, Moore and Gibson [10] who evaluated DPCM speech encoding with self-orthogonal convolutional codes, Reininger and Gibson [11] who studied backward adaptive prediction in DPCM speech coding along with high rate convolutional codes, and Goodman and Sundberg [12, 13] who develop embedded DPCM speech encoding and punctured convolutional codes.

Constrained joint source/channel coding is our third class of coders and is so named because the source coder and/or receiver are modified to account for the presence of a given noisy channel. In this class of coder we place source coders which have been (re-)optimized subject to a noisy channel constraint, such as the work by Kurtenbach and Wintz [14] on memoryless scalar quantization for discrete memoryless channels, Farvardin and Vaishampayan [15] on memoryless scalar quantizers and codeword assignments for the binary symmetric channel (BSC), Vaishampayan and Farvardin [16] on 2D-DCT image coders for the BSC, Kumazawa et al., on Linde, Buzo, Gray (LBG) vector quantization (VQ) of Gaussian sources for the BSC, and Chang and Donaldson [17] on the optimization of DPCM systems for noisy channel operation. Additional work on assigning binary codewords to quantizer outputs for noisy channels can be found in the Rydbeck and Sundberg [18], DeMarca and Jayant [19], and Zeger and Gersho [20].

Another subset in the class of constrained joint source/channel coders are those that use

some knowledge of the source or source coder properties to detect channel errors and compensate for their effects. In this group we include the work by Steele and Goodman, and Steele, Goodman, and McGonegal [21, 22] who detect errors in a speech coder output by monitoring the sample-to-sample differences in the reconstructed values and replacing those reconstructed values whose differences are too large by the output of a smoothing circuit, the work of Ngan and Steele [23], and Pitt, Swanson, and Yuen [24] who use similar ideas to motivate the development of a method to recover from errors in an image transmission system, the work of Reininger and Gibson [25] who use coefficients from neighboring blocks in a 2D-DCT image coding system to detect errors and smooth out their effects, and the work of Sayood and Borkenhagen [26, 27] who use redundancy in the coder output to perform sequence estimation. Hellman [28, 29] also suggests using the natural redundancy to correct errors in joint source/channel coding, and proposes a rate 1 catastrophic code to aid the process. The research described in this work is an extension of the work of Sayood and Borkenhagen [26, 27], and generalizes the approach of [21]–[25]. An earlier version of this work was presented in [30].

In the following section we describe the design criterion that is used for the various approaches presented in this paper. This is used in the following section to motivate some modification of existing source coder/convolutional coder design. The proposed modifications are evaluated using simulations, and the results of the evaluations are used to propose a class of non-binary convolutional encoders. Simulation results are presented which show that the proposed designs substantially outperform conventional systems (in the assumed scenario).

2 The Design Criterion

For a discrete memoryless channel (DMC), let the channel input alphabet be denoted by $A = \{a_0, a_1, \ldots, a_{M-1},\}$, and the channel input and output sequences by $Y = \{y_0, y_1, \ldots, y_{L-1}\}$ and $\hat{Y} = \{\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{L-1}\}$, respectively. If $\mathcal{A} = \{A_i\}$ is the set of sequences $A_i = \{\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,L-1}\}$, $\alpha_{i,k} \epsilon A$, then the optimum receiver (in the sense of maximizing the

probability of making a correct decision) maximizes $P[C]$, where

$$P[C] = \sum_{A_i} P[C|\hat{Y}]P[\hat{Y}]$$

This in turn implies that the optimum receiver maximizes $P[C|\hat{Y}]$. When the receiver selects the output to be A_k, then $P[C|\hat{Y}] = P[Y = A_k|\hat{Y}]$. Thus, the optimum receiver selects the sequence A_k such that

$$P[Y = A_k|\hat{Y}] \geq P[Y = A_i|\hat{Y}] \ \ \forall i$$

Noting that

$$P(Y|\hat{Y} = \frac{P(\hat{Y}|Y)P(Y)}{P(\hat{Y})}$$

and for fixed length codes $P(\hat{Y})$ is irrelevant to the receiver's operation, the optimal receiver maximizes $P(\hat{Y}|Y)P(Y)$. If we impose a first order Markov assumption on $\{y_i\}$, we can easily show that [31]

$$P(\hat{Y}|Y)P(Y) = \prod P(\hat{y}_i|y_i)P(y_i|y_{i-1}) \tag{1}$$

This result addresses the situation in which the source coder output (which is also the channel input sequence) contains redundancy. Using this result, we can design a decoder which will take advantage of dependence in the channel input sequence. The physical structure of the decoder can be easily obtained by examining the quantity to be maximized. The optimum decoder maximizes $P(\hat{Y}|Y)P(Y)$ or equivalently, $\log P(\hat{Y}|Y)P(Y)$, but

$$\log P(\hat{Y}|Y)P(Y) = \sum \log P(\hat{y}_i|y_i)P(y_i|y_{i-1}) \tag{2}$$

which is similar in form to the path metric of a convolutional decoder. Error correction using convolutional codes is made possible by explicitly limiting the possible codeword to codeword transitions, based on the previous code input and the coder structure. At the receiver the decoder compares the received data stream to the *a priori* information about the code structure. The output of the decoder is the sequence that is most likely to be the transmitted sequence. In the case where there is residual structure in the source coder output, the structure makes some sequences more likely to be the transmitted sequence,

given a particular received sequence. In other words, even when there is no structure being imposed by the encoder, there is sufficient residual structure in the source coder output that can be used for error correction. The structure is reflected in the conditional probabilities, and can be used via the path metric in (2) in a decoder similar in structure to a convolutional decoder. However, to implement this decoder we need to be able to compute the path metric.

Examining the branch metric, we see that it consists of two terms $\log P(\hat{y}_i|y_i)$ and $\log P(y_i|y_{i-1})$. The first term depends strictly on our knowledge of the channel. The second term depends only on the statistics of the source sequence. Therefore knowledge of both the channel and source statistics is necessary for implementing this path metric. In our simulations we have obtained the channel statistics by assuming that the channel is a binary symmetric channel with known probability of error. We have obtained the second term using a training sequence. The sequence used for testing the proposed approach is different from the training sequence.

In [26] we showed that the use of the decoder led to dramatic improvements under high error rate conditions. However at low error rates the performance improvement was from nonexistent to minimal. This is in contrast to standard error correcting approaches, in which the greatest performance improvements are at low error rates, with a rapid deterioration in performance at high error rates. In this work we combine the two approaches to develop a joint source channel codec which provides protection equal to the standard channel encoders at low error rates while also providing significant error protection at high error rates.

3 Convolutional Encoders and Joint Source/Channel Decoder

In [27] the output sequence of the source coder was taken as the sequence $\{y_i\}$. The received sequence $\{\hat{y}_i\}$ formed the input to the joint source/channel (JSC) decoder which was simply a viterbi decoder with a path metric similar to (2). The output of the JSC decoder was then passed to the source decoder. As mentioned above, this approach provided significant improvements only at high error rates. If we had used a standard convolutional encoder

with the source coder, this would have provided excellent error protection at low error rates (of course with an increase in the transmission rate). However in spite of the increase in transmission rate the convolutional coder still does not provide any protection at high error rates

The convolutional decoder uses the structure imposed by the encoder and the Hamming metric to provide error protection. The decoder does not use any of the residual structure from the source coder output. We can make use of the residual structure by noting that the path labels transmitted by the convolutional encoder comprise the channel input alphabet $\{y_i\}$. We can then use a training sequence to obtain the transition probabilities $\{P(y_i|y_{i-1})\}$, and an estimate of the channel error probability to obtain $\{P(\hat{y}_i|y_i)\}$. These can be used to compute the branch metric L

$$L = \log P(y_i|y_{i-1}) + \log P(\hat{y}_i|y_i) \tag{3}$$

which can be used instead of the Hamming metric in the decoder.

We simulated this approach using a two bit DPCM system as the source encoder. We used the two images shown in Figure 1 as the source. The USC Girl image was used for training (obtaining the requisite transition probabilities) and the USC Couple image was used for testing. The output of the DPCM system was encoded using a (2,1,3) convolutional encoder with connection vectors

$$g^{(1)} = 64 \quad g^{(2)} = 74 \tag{4}$$

The convolutional encoder was obtained from [32]. The performance of the different systems was evaluated using two different measures. One was the reconstruction signal-to-noise ratio (RSNR) defined as

$$RSNR = 10 \log_{10} \frac{\sum u_i^2}{\sum (u_i - \hat{u}_i)^2} \tag{5}$$

where u_i is the input to the source coder (source image) and $\hat{u}_i$ is the output of the source decoder (reconstructed image). The other performance measure was the decoded error probability. The received sequence was decoded using a standard convolutional decoder and the JSC decoder. A block diagram of the system is shown in Figure 2. The results are presented in Figure 3. Looking at the decoded error performance results in Figure 3a we

see that the performance seems to be about what we expected from the results of [27]: an improvement at high error rates for the system with the JSC decoder with some loss in performance at lower error rates. However, from Figure 3b we see that while the decoded error probability went down for the system with the JSC decoder at high error rates, there was no commensurate improvement in RSNR. In fact for probabilities of error below 0.15, there is a *decrease* in the RSNR for the system with the JSC decoder over a conventional system. While we had expected some slight disadvantage for the system with the JSC decoder at low error rates, the disadvantage here is not slight and is not restricted to low error rates.

To see why this happened, let us examine our assumptions. The use of the JSC decoder was predicated on the assumption that there was some structure in the source coder output. The unspoken assumption was that this structure would get translated to the channel coder output. In this particular example the source coder output is a sequence of two bit values. The channel coder takes the source coder output *one* bit at a time to generate the two bit convolutional coder output. It is highly unlikely that the sample-to-sample redundancy in the source coder output would translate to a bit-to-bit redundancy in the quantizer labels which could then be transferred to the convolutional encoder output. Thus our unspoken assumption is being violated, and the results in Figure 3 reflect this fact.

If the above hypothesis is indeed true, then the destruction of the structure in the source coder output could be prevented if we used a convolutional coder which uses an input wordlength k of two. In order to verify this hypothesis we used a $(4,2,1)$ convolutional coder which is equivalent to the $(2,1,3)$ coder in terms of rate and memory, but maps the two bit outputs from the source coder directly to the channel coder output. The connection vectors for this coder are [32]

$$g_1^{(1)} = 6 \quad g_1^{(2)} = 0 \quad g_1^{(3)} = 6 \quad g_1^{(4)} = 4$$

$$g_2^{(1)} = 0 \quad g_2^{(2)} = 6 \quad g_2^{(3)} = 4 \quad g_2^{(4)} = 2$$

The results for the $(4,2,1)$ coder, shown in Figure 4, seem to bear out our hypothesis. The decoded probability of error is uniformly better for the system with the JSC decoder. The same is true for the RSNR with an improvement of about 4 dB at $P(\epsilon) = 0.1$, and about a

6 dB improvement at $P(\epsilon) = 0.25$. This is a marked contrast to the results for the $(2,1,3)$ code shown in Figure 3b where there is a 4 dB *degradation* in the performance of the system with the JSC decoder at $P(\epsilon) = 0.1$.

The two rate 1/2 systems are compared in Figure 5. We have not included the results for the $(2,1,3)$ system with the JSC decoder to reduce clutter. Notice that while the conventional $(2,1,3)$ system is superior to the conventional $(4,2,1)$ system at low error rates, the $(4,2,1)$ system with the JSC decoder outperforms it as well.

The simulations were repeated with a rate 2/3 (3,2,2) convolutional coder with connection vectors

$$g_1^{(1)} = 7 \quad g_1^{(2)} = 1 \quad g_1^{(3)} = 4$$
$$g_2^{(1)} = 2 \quad g_2^{(2)} = 5 \quad g_2^{(3)} = 7$$

The results are shown in Figure 6. Notice that while there is some drop in performance for the system with the JSC decoder at low error rates, the overall performance is as expected. There is an improvement of about 6 dB at $P(\epsilon) = 0.1$.

In this section we have shown how the use of the residual redundancy in the source coder output can improve the performance of conventional source coder/convolutional coder systems. In order to make use of this redundancy we see that the channel coder input characteristics have to match the source coder output characteristics. In the next section we take this approach one step further and design channel coders with the specific goal of taking advantage of the redundancy in the source coder output. An additional advantage of the coders described in the next section is that there is an automatic match between the source coder output and the channel coder input.

4 A Modified Convolutional Encoder

Given that the preservation of the structure in the source coder output requires the channel coder input alphabet to have a one-to-one match with the generally nonbinary source coder, we propose a general nonbinary convolutional encoder (NCE) whose input alphabet has the requisite property.

Let x_n, the input to the NCE, be selected from the alphabet $A = \{0, 1, 2, ..., N-1\}$, and let y_n, the output alphabet of the NCE, be selected from the alphabet $S = \{0, 1, 2, ..., M-1\}$. Then the proposed NCEs can be described by the following mappings

1. $M = N^2$; $y_n = Nx_{n-1} + x_n$

The number of bits required to represent the output alphabet using a fixed length code is

$$\lceil \log_2(M) \rceil = \lceil \log_2(N^2) \rceil = \lceil 2 \log_2(N) \rceil$$

Therefore in terms of rate, this coder is equivalent to a rate 1/2 convolutional encoder. The encoder memory in bits is $2\lceil \log_2(N) \rceil$ as each output value depends on two input values.

As an example, consider the situation when $N = 4$. Then $A = \{0, 1, 2, 3\}$ and $S = \{0, 1, 2, ..., 15\}$. Given the input sequence x_n : 0 1 3 0 2 1 1 0 3 3 and assuming the encoder is initialized with zeros, the output sequence will be y_n : 0 1 7 12 2 9 5 4 3 15.

The encoder memory is four bits. Notice that while the encoder output alphabet is of size N^2, at any given instant the encoder can only emit one of N different symbols as should be the case for a rate 1/2 convolutional encoder. For example if $y_{n-1} = 0$, then y_n will take on a value from $\{0, 1, 2, ..., (N-1)\}$. In general, given a value for y_{n-1}, y_n will take on a value from $\{\alpha N, \alpha N + 1, \alpha N + 2, ..., \alpha N + N - 1\}$, where $\alpha = y_{n-1}(mod N)$. This structure can be used by the decoder to provide error protection. The encoder is shown in Figure 7a.

2. $M = N^3$; $y_n = N^2 x_{n-2} + Nx_{n-1} + x_n$

This encoder is equivalent to a rate 1/3 convolutional encoder with an encoder memory in bits of $3\lceil \log_2(N) \rceil$. Given the same input as the previous example, the output alphabet for the NCE is

$$S = \{0, 1, 2, ..., 63\}$$

and the output sequence for the same input sequence is

$$y_n : \ 0\ 1\ 7\ 28\ 50\ 9\ 37\ 20\ 19\ 15$$

The encoder memory is six bits. In this case even though the encoder output alphabet is of

size N^3, at any instant the encoder can only emit one of N symbols. In general, given a value for y_{n-1}, y_n will take on a value from $\{\beta N, \beta N+1, ..., \beta N+N-1\}$, where $\beta = y_{n-1}(mod N^2)$. A block diagram of the encoder is shown in Figure 7b.

3. $M = N^3$

$$y_n = N^2 x_{2n} + N x_{2n-1} + x_{2n-2}$$

The final encoder we consider is equivalent to a rate 2/3 convolutional coder. Notice that while the input output relationship looks similar to a rate 1/3 encoder, we generate one output for every two inputs. Thus, while the number of bits needed to represent one letter from the output alphabet is three times the bits needed to represent a letter from the input alphabet, the rate is 2/3 because two input letters are represented by a single output letter. This coder could be viewed as a rate 2/3 punctured nonbinary convolutional coder. Again, assuming a value of 4 for N, the output alphabet is of size 64, and for the input sequence used previously, the output sequence is y_n : 0 52 35 22 49 3.

The encoder memory is again 6 bits. The rate of the encoder can also be inferred from the fact that while the encoder output alphabet is of size N^3, at any instant the encoder can transmit one of N^2 (instead of N) symbols. Given a value for y_{n-1}, y_n can take on a value from the alphabet $\{\gamma N^2, \gamma N^2 + 1, ..., \gamma N^2 + (N^2 - 1)\}$ where $\gamma = y_{n-1}(mod N)$. A block diagram of the encoder is shown in Figure 7c.

All of these encoders can be designed for any value of N. Furthermore, their input and output alphabets as described above can easily be seen as indices to tables of codewords. We will exploit this latter property in the next section for allocating codewords to the NCE outputs.

4.1 Binary Encoding of the NCE Output

We will make use of the residual structure in the source coder output (which is preserved in the NCE output) at the receiver. However, we can also make use of this structure in selecting binary codes for the NCE output. An intelligent assignment of binary codes can

improve the error correcting performance of the system as can be seen from the following example.

Let N be 2, and let us use the rate 1/2 NCE. In this case if $y_n = 0$, y_{n+1} cannot be 2 or 3, because $y_n = 0$ means $x_n = 0$, and $y_{n+1} = 2$ or 3 means $x_n = 1$. Thus a decoded sequence cannot have 2 or 3 following 0.

Let us assign fixed length codewords to the NCE outputs as

$$0 : 00,\ 1 : 01,\ 2 : 10,\ 3 : 11$$

Now suppose the transmitted sequence was the all zero sequence, the metric used was the Hamming distance, and the received sequence is 00001000000000; that is, there is an error in the fifth bit. If the receiver decoded the first four bits as 0, 0 then it cannot decode the fifth and sixth bits as 2 for the reason noted above. The only two options are decoding them as 0 or 1. If we decoded them as 0, we could continue decoding the rest of the sequence as $0, 0 \ldots$, and the Hamming distance between the received and decoded sequence would be one. If we decoded them as 1, we would have to decode the next set of two bits as 2 or 3 because 0 cannot follow 1. Decoding as 2 gives the smallest Hamming distance so we decode the seventh and eighth bit as 2. This gives a total Hamming distance of two for the incorrect path. Thus the receiver will select the correct path (the path with the smallest Hamming distance). If the assignment had been chosen as

$$0 : 00;\ 1 : 11;\ 2 : 10;\ 3 : 01$$

then the Hamming distance for the closest incorrect path would have been three instead of two.

When each allowable sequence is equally likely, there is little reason to prefer one particular assignment over others. However, when certain sequences are more likely to occur than others, it would be useful to make assignments which increase the 'distance' between likely sequences. While, for small alphabets it is a simple matter to assign the optimum binary codewords by inspection, this becomes computationally impossible for larger alphabets. We use a rather simple heuristic which, while not optimal, provides good results.

The number of M bits codewords that have to be assigned are exactly 2^M. Our strategy is

therefore to try to maximize the Hamming distance between codewords that are likely to be mistaken for one another.

First we obtain a partition of the alphabet based on the fact that given a particular value for y_{n-1}, y_n can only take on values from a subset of the full alphabet. To see this, consider the rate 1/2 NCE; then the alphabet S can be partitioned into the following sub-alphabets:

$$S_0 = (0, 1, 2, \ldots, N-1)$$

$$S_1 = (N, N+1, N+2, \ldots, 2N-1)$$

$$\vdots$$

$$S_{N-1} = (N(N-1), N(N-1)+1, N(N-1)+2, \ldots, N^2-1)$$

where the encoder will select letters from alphabet S_j at time n if $j = y_{n-1} (\mathrm{mod} N)$. Now for each sub-alphabet we have to pick N codewords out of M $(= N^2)$ possible choices. We first pick the sub-alphabet containing the most likely letter. The letters in the sub-alphabet are ordered according to their probability of occurrence. We assign a codeword a from the list of available codewords to the most probable symbol. Then, assign the complement of a to the next symbol on the list. Therefore the distance between the two most likely symbols in the list is $K = \lceil \log_2 M \rceil$ bits. We then pick a codeword b from the list which has maximum distance from a such that the Hamming distance from a and the Hamming distance from the complement of a differ by at most one. We assign it and its complement to the next two elements on the list. This process is continued until all letters in the subalphabet have a codeword assigned to them.

As an example, consider the case where $N = 4$. The partitions are

$$S_0 = (0, 1, 2, 3)$$

$$S_1 = (4, 5, 6, 7)$$

$$S_2 = (8, 9, 10, 11)$$

$$S_3 = (12, 13, 14, 15)$$

Assuming that 0 is the most probable symbol, we start by assigning codewords to the S_0 sub-alphabet. Suppose

$$P(0) \geq P(3) \geq P(1) \geq P(2)$$

We first pick a 4 bit codeword for 0 as **0000**. The next most probable symbol in this sub-alphabet is 3; therefore the codeword for 3 is the complement of the codeword for 0; 3:**1111**. The codeword for 1 is at a Hamming distance of two from the codeword for 0 and the codeword for 3. The codeword 0011 satisfies this requirement; therefore the codeword for 1 is **0011** and the codeword for 2 is **1100**. Suppose the next symbol which is close in probability to the symbol 0 is 4. We select the sub-alphabet containing that symbol which is S_1. To the symbol 4 we assign a codeword from the list of unassigned codewords which is furthest from the codeword for 0. There are several possibilities for this; we pick **1110**. We then follow the same procedure for the S_1 sub-alphabet. Continuing in this manner we get the assignments shown in Table 1.

4.2 Simulation Results

The proposed nonbinary convolutional encoders were simulated using the same setup as was used in the previous simulations. The binary assignments were made using the statistics of the training image which again was the USC Girl image. The test image was once more the USC Couple image. The simulation results are presented in Figures 8 and 9.

In Figure 8 the performance of the rate 1/2 NCE is plotted alongside the results for the (2,1,3) coder and the (4,2,1) coder. For both situations we used the system with the JSC decoder. Recall that in the previous simulations the (4,2,1) coder with the JSC decoder substantially outperformed all other rate 1/2 systems. From the results in Figure 8 we can see that the rate 1/2 NCE substantially outperforms the (4,2,1) coder with the JSC decoder. Comparing these results against the original results in Figure 3 for the conventional (2,1,3) coder we see that the rate 1/2 NCE provides impressive gains: at $P(\epsilon) = 0.1$ the gain of the NCE over the conventional (2,1,3) system is about 8 dB while at $P(\epsilon) = 0.25$, the gain is about 12 dB! Even more important is the fact that the performance of the rate 1/2 NCE

is relatively flat over a very wide range of channel conditions. The RSNR at $P(\epsilon) = 0.1$ is less than 1 dB below the value obtained under noiseless channel conditions. Over the entire range of channel error probabilities from 0 to 0.25, the RSNR drops slightly more than 4 dB

The performance of the rate 2/3 NCE, while not as impressive as the rate 1/2 NCE on an absolute scale, is still excellent in a relative sense. The performance improvement over the conventional (3,2,2) coder at $P(\epsilon) = 0.1$ is again about 8 dB. In fact at that error rate the rate 2/3 NCE outperforms the conventional rate 1/2 systems.

5 Conclusion

In this paper we have presented two ways of using the structure in the source coder output for forward error correction. The first approach simply modifies the decoder in a conventional source coder/convolutional coder system to take advantage of the residual redundancy. This approach would be especially useful in situations where a conventional system was already in place and a change in channel characteristics or transmission requirements required that the system be 'updated'. The modifications in this case would be relatively modest and would only need to be performed at the decoder. The simulation results show improvements of more than 6 dB at high error rates. The second approach involves a new design of the channel encoder. The simulation results pertaining to this design show excellent performance over a wide range of channel error probabilities, from 0.0 to 0.25. The performance improvements range as high as 12 dB at high error rates. This design might be especially useful in the codecs designed for the mobile radio communication channel.

An issue that has not been considered in this paper is the effect of mismatch between the actual and assumed channel statistics. Another important area of future research is the development of a theoretical measure analogous to d_{free} for use in predicting and classifying performance of the coders developed in this paper.

References

[1] C. E. Shannon. The Mathematical Theory of Communication. *Bell Systems Technical Journal*, 28:379–423, October 1949.

[2] J.L. Massey. Joint Source and Channel Coding. In J.K. Skwirzynski, editor, *Communication Systems and Random Process Theory*, pages 279–293. Sijthoff and Nordhoff, 1978.

[3] T.C. Ancheta Jr. *Joint Source Channel Coding.* PhD thesis, University of Notre Dame, August 1977.

[4] J.G. Dunham and R.M. Gray. Joint Source and Channel Trellis Encoding. *IEEE Transactions on Information Theory*, IT-27:516–519, July 1981.

[5] E. Ayanoğlu and R.M. Gray. The Design of Joint Source and Channel Trellis Waveform Coders. *IEEE Transactions on Information Theory*, IT-33:855–865, November 1987.

[6] J.W. Modestino and D.G. Daut. Combined Source Channel Coding of Images. *IEEE Transactions on Communications*, COM-27:1644–1659, November 1979.

[7] J.W. Modestino, D.G. Daut, and A.L. Vickers. Combined Source Channel Coding of Images Using the Block Cosine Transform. *IEEE Trans. Commun.*, COM-29:1262–1274, September 1981.

[8] J.W. Modestino, V. Bhaskaran, and J.B. Anderson. Tree Encoding of Images in the Presence of Channel Errors. *IEEE Transactions on Information Theory*, IT-27:667–697, November 1981.

[9] D. Comstock and J.D. Gibson. Hamming Coding of DCT Compressed Images Over Noisy Channels. *IEEE Trans. Commun.*, COM-32:856–861, July 1984.

[10] C.C. Moore and J.D. Gibson. Self-Orthogonal Convolutional Coding for the DPCM-AQB Speech Encoder. *IEEE Trans. Commun.*, COM-32, August 1984.

[11] R.C. Reininger and J.D. Gibson. Backward Adaptive Lattice and Transversal Predictors in ADPCM. *IEEE Transactions on Communications*, COM-33:74–82, January 1985.

[12] D.J. Goodman and C.E. Sundberg. Combined Source and Channel Coding for Variable Bit-Rate Speech Transmission. *Bell System Technical Journal*, 62:2017–2036, September 1983.

[13] D.J. Goodman and C.-E. Sundberg. Transmission Errors and Forward Error Correction in Embedded Differential PCM. *Bell Syst. Tech. J.*, 62:2735–2764, November 1983.

[14] A.J. Kurtenbach and P.A. Wintz. Quantizing for Noisy Channels. *IEEE Trans. Commun. Technol.*, COM-17:291–302, April 1969.

[15] N. Farvardin and V. Vaishampayan. Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding. *IEEE Trans. Inform. Theory*, IT-33:827–838, November 1987.

[16] V.A. Vaishampayan and N. Farvardin. Optimal Block Cosine Transform Image Coding for Noisy Channels. *IEEE Transactions on Communications*, COM-38:327–336, March 1990.

[17] K.-Y. Chang and R.W. Donaldson. Analysis, Optimization, and Sensitivity Study of Differential PCM Systems Operating on Noisy Communication Channels. *IEEE Trans. Commun.*, COM-20:338–350, June 1972.

[18] N. Rydbeck and C.-E. Sundberg. Analysis of Digital Errors in Nonlinear PCM Systems. *IEEE Transactions on Communications*, COM-24:59–65, January 1976.

[19] J.R.B. DeMarca and N.S. Jayant. An Algorithm for Assigning Binary Indices to the Codevectors of a Multi-dimensional Quantizer. In *Proceedings ICC '87*, pages 1128–1132. IEEE, 1987.

[20] K. Zeger and A. Gersho. Vector Quantizer Design for Memoryless Noisy Channels. In *Proceedings ICC '88*, pages 1593–1597. IEEE, June 1988.

[21] R. Steele and D.J. Goodman. Detection and Selective Smoothing of Transmission Errors in Linear PCM. *Bell System Technical Journal*, 56:399–409, March 1977.

[22] R. Steele, D.J. Goodman, and C.A. McGonegal. A Difference Detection and Correction Scheme for Combatting DPCM Transmission Errors. *IEEE Trans. Commun.*, COM-27:252–255, January 1979.

[23] K.N. Ngan and R.Steele. Enhancement of PCM and DPCM Images Corrupted by Transmission Errors. *IEEE Trans. Commun.*, COM-30:257–269, January 1982.

[24] G.H. Pitt III, L. Swanson, and J.H. Yuen. Image Statistics Decoding for Convolutional Codes. Technical Report TDA Progress Report 42-90, JPL, April-June 1987.

[25] R.C. Reininger and J.D. Gibson. Soft Decision Demodulation and Transform Coding. *IEEE Trans. Commun.*, COM-31:572–577, April 1983.

[26] K. Sayood and J.C. Borkenhagen. Utilization of Correlation in Low Rate DPCM Systems for Channel Error Protection. In *Proceedings IEEE International Conference on Communications*, pages 1888–1892. IEEE, June 1986.

[27] K. Sayood and J.C. Borkenhagen. Use of Residual Redundancy in the Design of Joint Source/Channel Coders. *IEEE Transactions on Communications*, 39:838–846, March 1991.

[28] M. E. Hellman. On Using Natural Redundancy for Error Detection. *IEEE Transactions on Communications*, COM-22:1690–1693, October 1974.

[29] M. E. Hellman. Convolutional Source Encoding. *IEEE Transaction on Information Theory*, IT-21:651–656, November 1975.

[30] K. Sayood, F. Liu, and J.D. Gibson. A Joint Source/Channel Coder Design. In *Proceedings International Conference on Communications*, pages 727–731. IEEE, May 1993.

[31] N. Phamdo and N. Farvardin. Optimal Detection of Discrete Markov Sources over Discrete Memoryless Channels – Applications to Combined Source Channel Coding. To appear IEEE Transactions on Information Theory.

[32] S. Lin and D.J. Costello. *Error Control Coding.* Prentice-Hall, 1983.

List of Figures

a) USC Girl

b) USC Couple

Figure 1. a) Training Image; b) Test Image

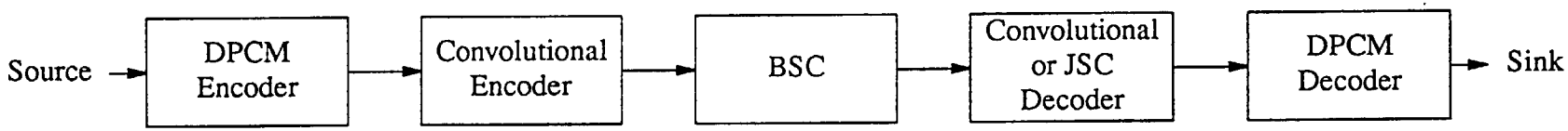


Figure 2. System block diagram.

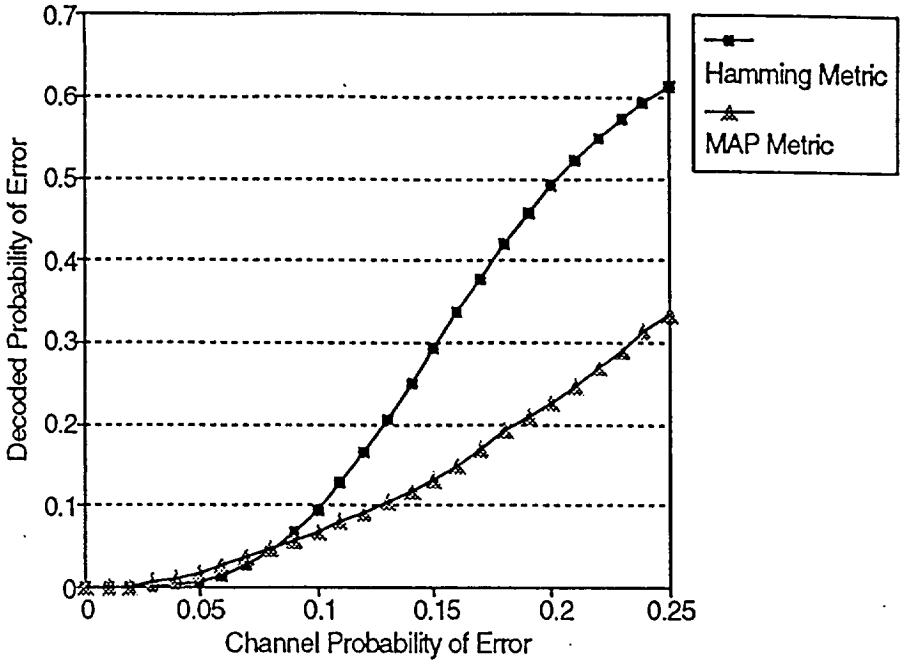


Fig. 3a. Error performance of (2,1,3) coder with conventional and JSC decoders

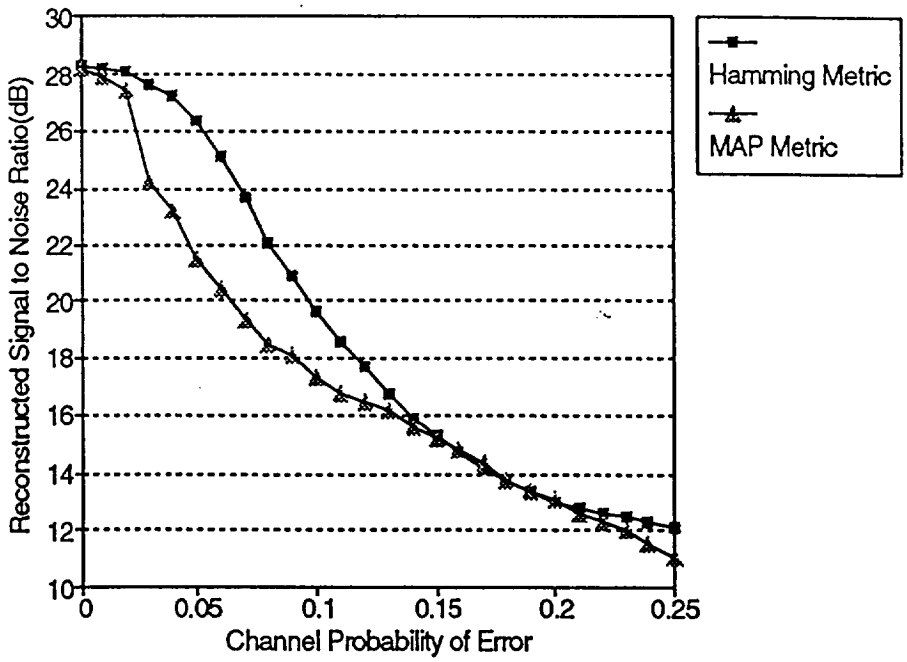


Fig. 3b. RSNR performance of (2,1,3) coder with conventional and JSC decoders

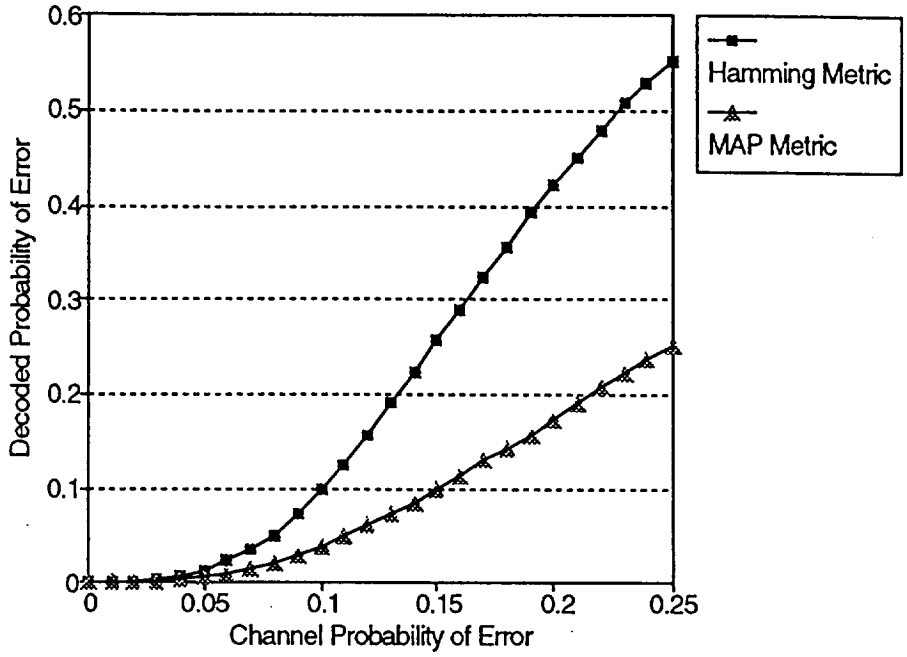


Fig. 4a. Error performance of (4,2,1) coder with conventional and JSC decoders

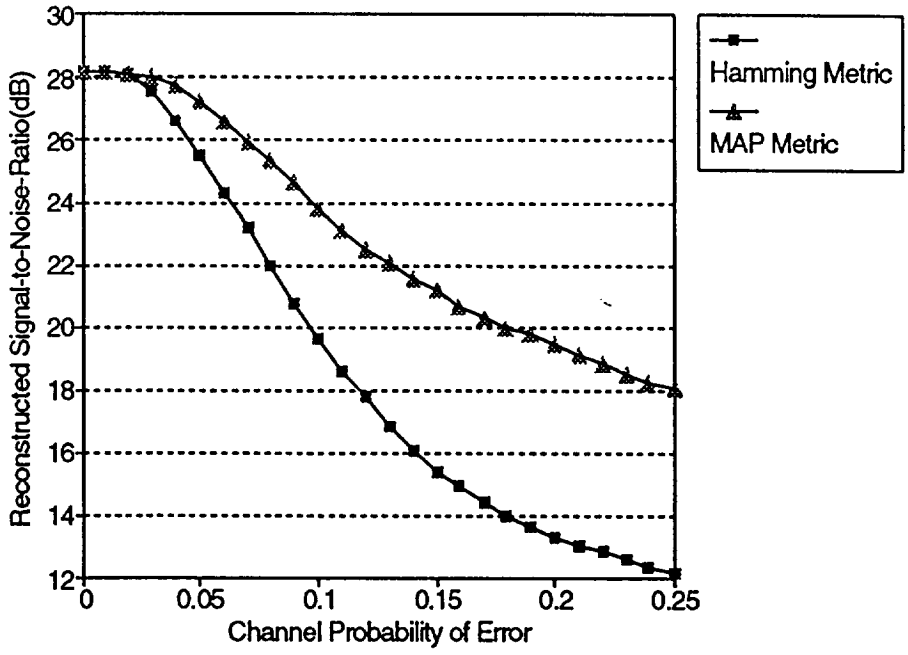


Fig. 4b. RSNR performance of (4,2,1) coder with conventional and JSC decoders

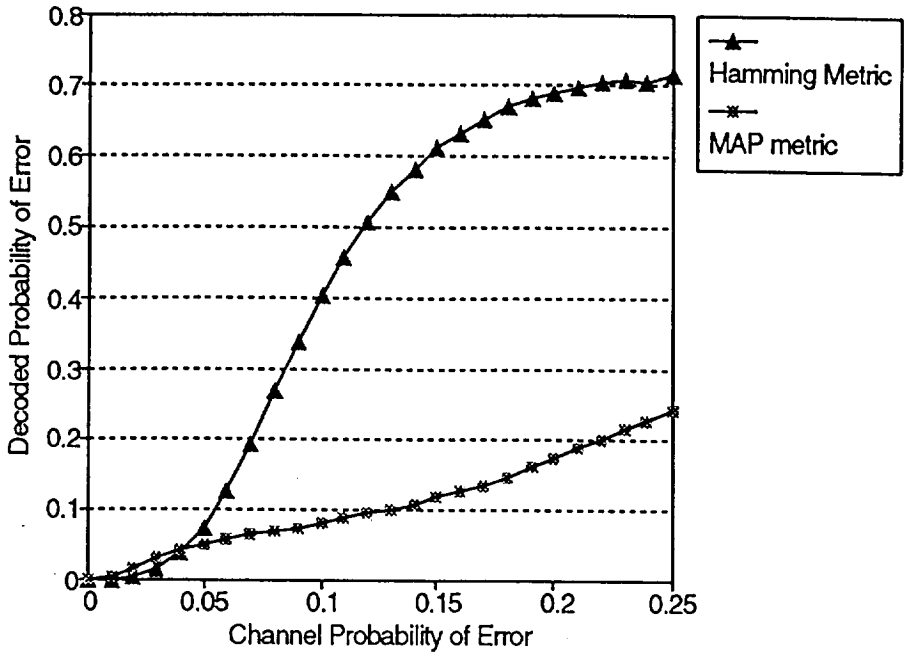


Fig. 6a. Error performance of (3,2,2) coder with conventional and JSC decoders

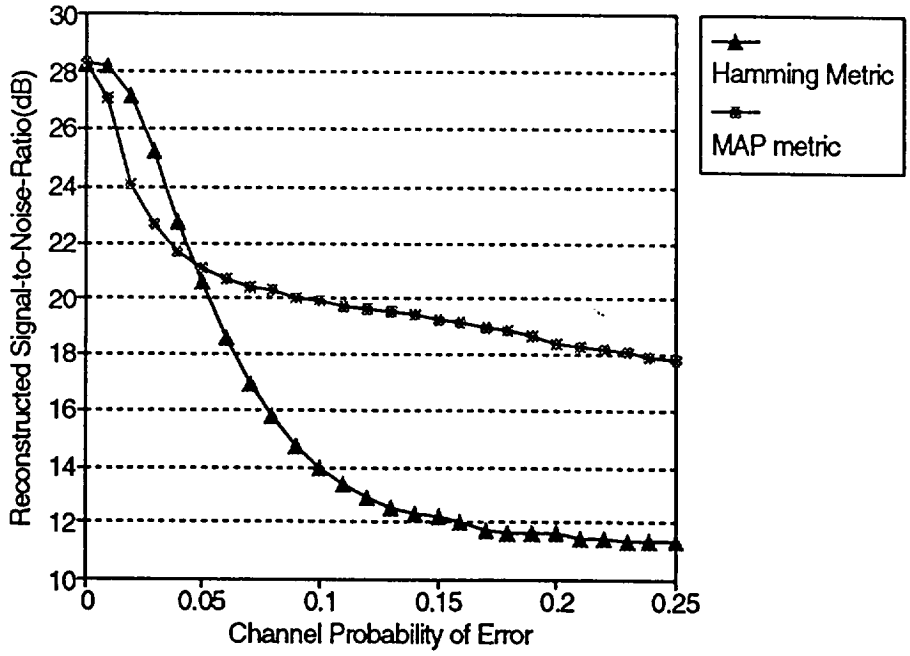
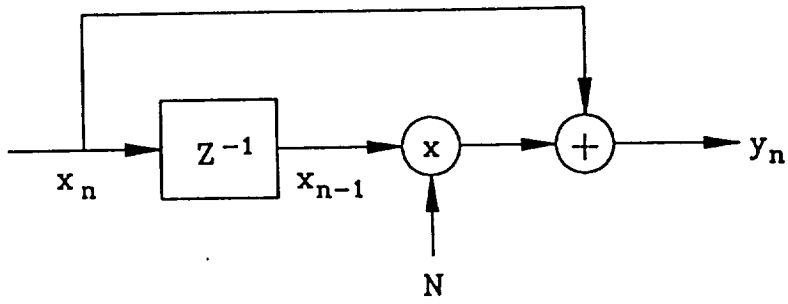
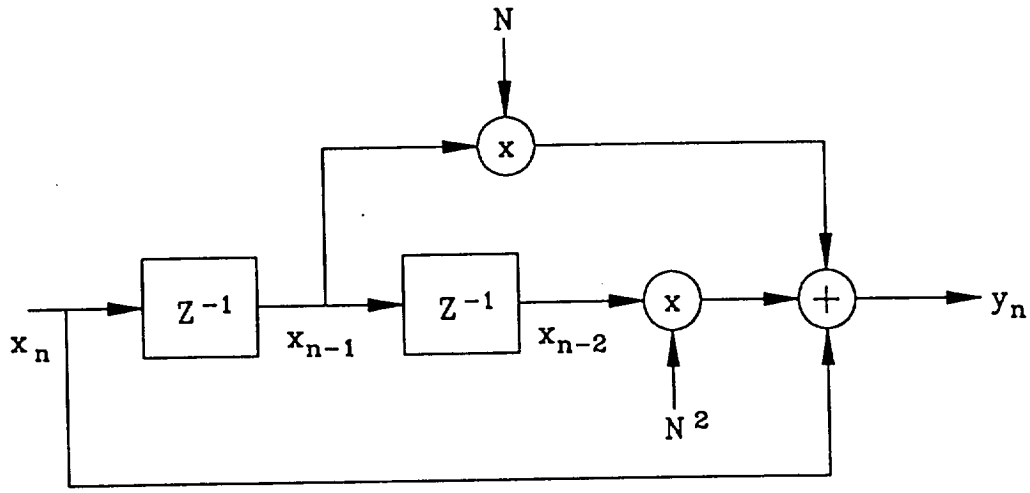


Fig. 6b. RSNR performance of (3,2,2) coder with conventional and JSC decoders



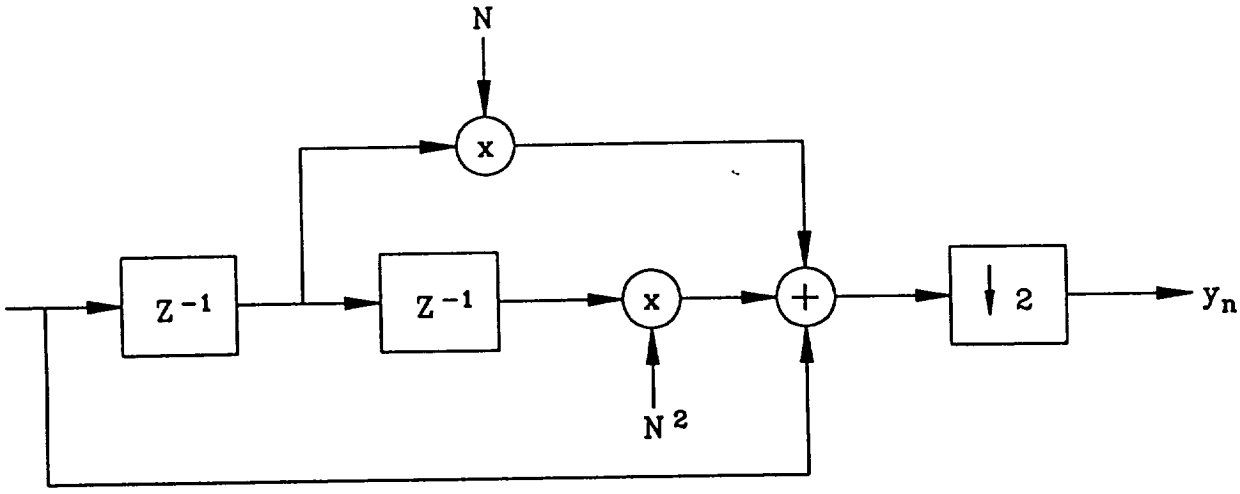
Rate 1/2 Nonbinary Convolutional Encoder

(a)



Rate 1/3 Nonbinary Convolutional Encoder

(b)



Rate 2/3 Nonbinary Convolutional Encoder

(c)

Figure 7. Proposed Nonbinary Convolutional Encoders

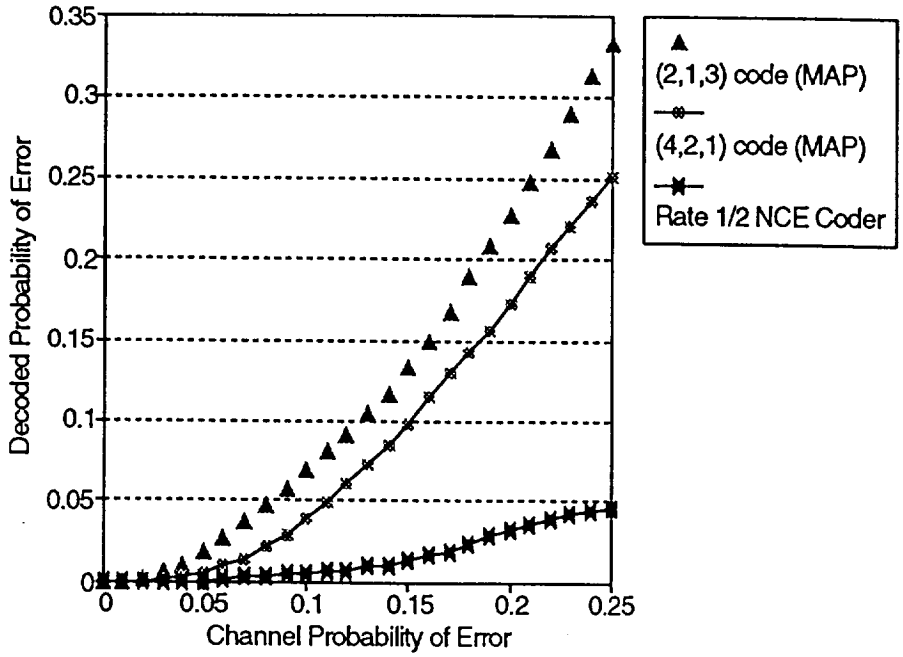


Fig. 8a. Error performance of rate 1/2 coders with JSC decoders

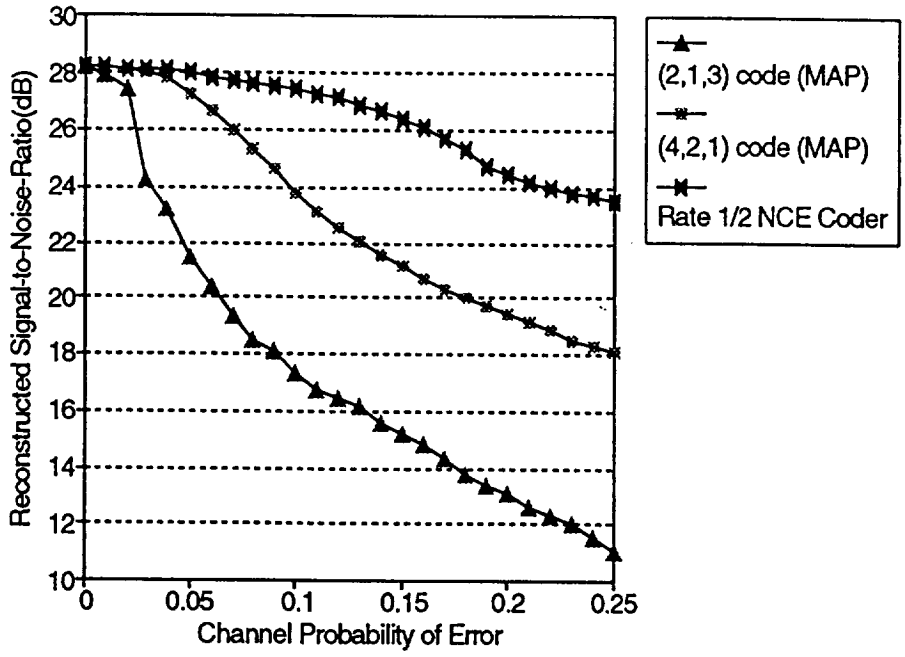


Fig. 8b. RSNR performance of rate 1/2 coders with JSC decoders

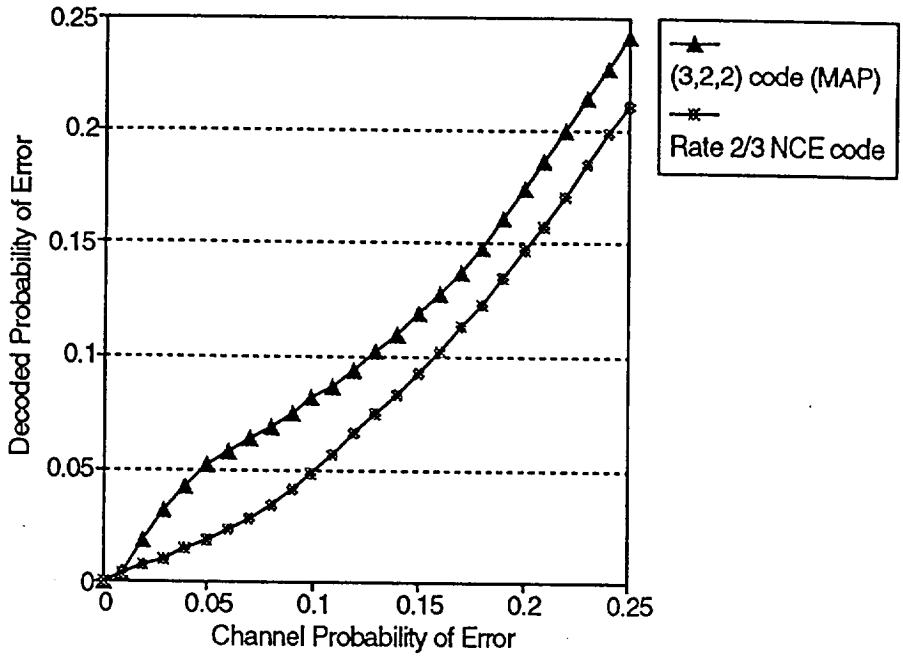


Fig. 9a. Error performance of rate 2/3 coders with JSC decoders

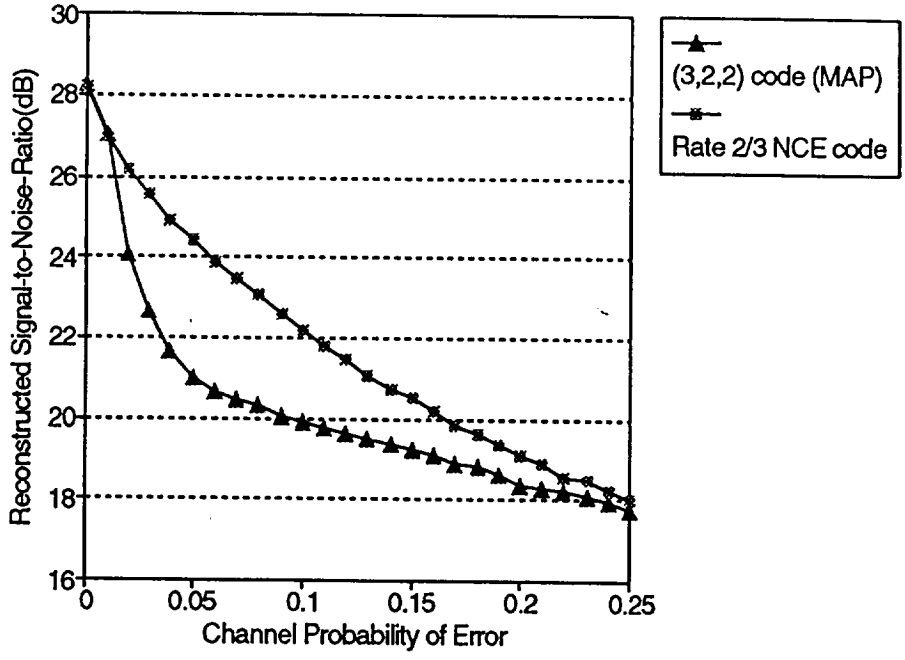


Fig. 9b. RSNR performance of rate 2/3 coders with JSC decoders

Table 1: Codeword Assignments

Symbol	Code	Symbol	Code
0	0000	8	1011
1	0011	9	0111
2	1100	10	0100
3	1111	11	1000
4	1110	12	0101
5	1101	13	1001
6	0001	14	1010
7	0010	15	0110